Using geometric algebra to represent and interpolate tool poses

Cripps, Robert; Mullineux, Glen

DOI:
10.1080/0951192X.2015.1034783

International Journal of Computer Integrated Manufacturing
Vol. 00, No. 00, Month 200x, 1–30

RESEARCH ARTICLE

Using geometric algebra to represent and interpolate tool poses

Robert J. Cripps[a] and Glen Mullineux[b]*
[a] *Department of Mechanical Engineering, University of Birmingham, Birmingham, UK* ; [b] *Department of Mechanical Engineering, University of Bath, Bath, UK*
(*Received 00 Month 200x; final version received 00 Month 200x*)

In conventional computer aided manufacturing techniques, a tool path is sent to the machine tool controller as a sequence of precision points. This means that errors are introduced when the controller interpolates between the points to recover the intended path. For 4- and 5-axis machining, a sequence of precision poses (positions and orientations) is needed. The use of geometric algebra in representing poses is discussed. The form of algebra used can represent rigid-body transforms exactly and can interpolate, in a natural way, between two or more poses using spherical linear interpolation (slerp). Sequences of poses from free-form surfaces are investigated. Tool motions are generated and compared with tool paths derived from purely positional information. It is found that the motion paths more naturally follow the original surface so that the interpolation errors are smaller.

Keywords: tool poses; numerical control; interpolation; geometric algebra

1. Introduction

The conventional means for passing from a computer aided design (CAD) model to a physical, manufactured part is to use the techniques of computer aided manufacture (CAM). Appropriate tool paths or curves are established using the CAD model. The aim is to reduce the amount of under- and over-cutting so that, if the tool curves are followed exactly, the resultant physical component is a sufficiently accurate reproduction of the CAD part. These curves are sent to the controller of the machine tool as sequences of *precision points*. The controller then interpolates between the precision points and drives the cutter to follow these to recreate the required curves.

It is possible to model some physical aspects of the machining process and compensate for these. This includes: the effects as feed-rates, tool vibration and work piece deformation; and the dynamics of the tool or the machine itself.

*Corresponding authors. Email: g.mullineux@bath.ac.uk

ISSN: 0951-192x print/ISSN 1362-3052 online

DOI: 10.1080/0951192YYxxxxxxxx
http://www.informaworld.com

However, if such physical effects are ignored, there are two main sources of error (Bhuiya and Tutunea-Fatan 2013). The first is the *interpolation error* whereby a curve recreated by the controller is not the same as the original from which the precision points were sampled. The second is the *control error* where the controller fails to drive the cutter exactly along its interpolated curve. Some of these problems are due to the limitations in the form of the G-code instructions. One way to reduce errors is to enhance these instructions to allow all the points on a tool path across a surface to be specified exactly (Koren and Lin 1995). Similarly, the STEP-NC standard can be enhanced to improve communication with the CNC system (Liang and Li 2013). Another way of reducing errors while maintaining commonly available NC facilities is to increase the density of precision points along the required curves. However this has the effect of slowing down the manufacturing process.

An alternative approach is to consider the form of the curves. It is possible to relate machining parameters such as cutter geometry and the step-over distance and hence ensure that an optimal choice of precision points is made (Zhu *et al.* 2012, Senatore *et al.* 2012, Liu *et al.* 2010). Similarly it is possible to consider where machining errors are likely to occur and optimize the curves themselves to reduce these (Beudaert *et al.* 2011). Other approaches have attempted to take advantage of the geometric properties of a curve, represented in Bézier or, more generally, NURBS form. This form allows simply evaluation in terms of position, velocity (first derivative), acceleration (second derivative) and jerk (third derivative) along the curve (Rauch *et al.* 2012, Annoni *et al.* 2012, Li *et al.* 2012, Zhang *et al.* 2011, Lei and Wang 2009, Mohan *et al.* 2008). The higher these derivative values, the more difficult it is for the controller to perform its interpolation as expected. (The higher derivatives can also help deal with non-geometric issues. For example, it is helpful to limit jerk and to ensure kinematic continuity between tool-path segments (Heng and Erkorkmaz 2010).)

The fact that these methods rely on existing proprietary controllers is perhaps a drawback, since in practice commercial confidentiality means that it is not clear precisely how a controller is working. Thus generic solutions are intractable.

The concept of working with precision points arose when early CAM systems were based around the use of $2\frac{1}{2}$- and 3-axis machining. However, working with precision points alone places more emphasis on the controller to determine the best interpolation path. If more information relating to the path is available, a curve more compatible with the controller may be easier to determine. Further, the advent of 4- and 5-axis machining and their extra degrees of freedom, has given rise to the need for information relating not only to the relative position of a cutter tool with respect to the work piece, but also its relative orientation. What is required are *precision poses* for the cutter which specify not only position but also orientation (with respect to the work-piece).

In performing interpolation, the interest is no longer in forming a curve between two points. Instead some higher form of interpolation is required as in (Bhuiya and Tutunea-Fatan 2013, Shen *et al.* 2011). What is considered in this paper is the generation of a motion between two poses.

There are a number of existing ways of describing and generating three-dimensional motions (Röschel 1998, Bottema and Roth 1979). One approach is to deal with the motion curves of individual points within a body and then combine these for the body itself (Hofer *et al.* 2004). This requires compensation to be made to avoid distortion of the body using matrix algebra (matrices and vectors). However, these have problems associated with robustness: for example, while every transform can be represented by a matrix, not every matrix represents a proper transform, and hence numerical errors in

calculations can cause corruption.

The growth in the complexity of computer games has seen a renewed interest in the use of quaternions to represent rotations (Leeney 2009, Fang *et al.* 1998, Shoemake 1985). These have been extended to form double and dual quaternions which can be used to describe motions (Wu and You 2010, Jin and Ge 2010, Purwar *et al.* 2008, Purwar and Ge 2005, Ahlers and McCarthy 2001), and handle kinematic control problems (Sariyildiz and Temeltas 2012, Wang *et al.* 2012, Akyar 2008).

An approach which is a natural extension of quaternions is the use of geometric algebra (Perwass 2009, Dorst *et al.* 2007, González Calvet 2007) and the proposed method presented in this paper uses a geometric algebra to represent tool path motions. This allows any rigid-body transform to be represented exactly (Cripps and Mullineux 2012, Mullineux and Simpson 2011), meaning that it can deal with the translational and rotational elements of a tool pose and can do so in a single theoretical framework. This is one of the aspects of the geometric algebra formulation that makes it very attractive for a number of applications. As well as allowing translations and rotations to be handled together, the algebra also allows ideas from free-form curve manipulation to be extended in a natural way to describe motions.

The aim of this paper is to consider the feasibility of using geometric algebra to represent tool motions. Of particular interest is the form of the geometry of the motion. Other factors, such as speed, can be controlled directly through the CNC controller.

To establish the feasibility of the geometric algebra, two forms of interpolation are compared. The first is *curve interpolation* (CI) in which a number of precision points are specified and a curve is formed passing through them. The second is *motion interpolation* (MI) in which precision poses are given and a motion through them is formed.

The remainder of the paper is organised as follows. An overview of the form of geometric algebra used here is given in section 2 and its representation of rigid-body transforms is discussed in section 3. More details on related approaches using quaternions and other formulations of geometric algebra are presented in the appendix for completeness.

Section 4 illustrates that the algebra is capable of generating the motions needed for typical MI between given tool poses. Section 5 looks at the construction of motions through precision poses, the number of degrees of freedom involved and the requirement to ensure that the correct choice of motion between a pair of poses is made.

Section 6 presents a numerical comparison between CI and MI motions constructed from free-form surfaces. It is seen that the interpolation error associated with MI is typically the smaller, suggesting that such motions better follow the free-form nature of the original surface. Finally, conclusions are drawn in section 7.

2. Overview of geometric algebra $\mathcal{G}_4$

There are a number of ways to define or construct geometric (or Clifford) algebras (Perwass 2009, Dorst *et al.* 2007, González Calvet 2007). The approach adopted here is that described in detail in (Cripps and Mullineux 2012, Mullineux and Simpson 2011) which creates an algebra called $\mathcal{G}_4$. Some alternative approaches are given in the appendix. The purpose of this section is to provide an overview of $\mathcal{G}_4$ and its properties. More details are given in the works last cited above.

The typical element of the algebra has the form

$$\begin{aligned} A &= \sum_{\sigma} a_\sigma \, e_\sigma \\ &= a_\phi + a_0 e_0 + a_1 e_1 + a_2 e_3 + a_3 e_3 \\ &\quad + a_{01} e_{01} + a_{02} e_{02} + a_{03} e_{03} \\ &\quad + a_{12} e_{12} + a_{13} e_{13} + a_{23} e_{23} \\ &\quad + a_{012} e_{012} + a_{013} e_{013} + a_{023} e_{023} + a_{123} e_{123} \\ &\quad + a_\omega e_\omega \end{aligned} \tag{1}$$

where the sum is over subsets σ of the set $\{0, 1, 2, 3\}$, the e_σ are basis elements with e_ϕ identified with 1, the unit scalar, and ω used to denote e_{0123}, and $a_\sigma \in \mathrm{R}$.

Multiplication (which is non-commutative) is defined by using the following rules

$$e_{ij} = e_i e_j = -e_j e_i = -e_{ji} \qquad \text{for } i \neq j \tag{2}$$

$$e_1^2 = e_2^2 = e_3^2 = 1 \tag{3}$$

$$e_0^2 = \varepsilon^{-1} \tag{4}$$

where ε is a symbol representing a small (positive) real number. This means that the coefficients in equation (1) are real power series in ε.

The basis element $\omega = e_{0123}$ is called *the pseudo-scalar*. More generally, any element of the form $\alpha + \varepsilon\beta\omega$ where $\alpha, \beta \in \mathrm{R}$ are scalars is called *a pseudo-scalar*. The set of pseudo-scalars is closed under multiplication.

The *grade* of the basis element e_σ is the size of the subset σ. The typical element A in equation (1) has grade i if it is a combination only of basis elements of grade i. Elements of grade 1 are called *vectors*; those of grade 2 are *bivectors*, and those of grade 3 are *trivectors*. The only elements of grades zero and 4 are scalars and scalar multiples of the pseudo-scalar respectively. The definition of multiplication ensures that the product of two elements of even grade or of odd grade has even grade; and the product of two elements with different parities has odd grade.

The typical even-grade element of the algebra has the form

$$\begin{aligned} \sum_{\sigma} a_\sigma \, e_\sigma = a_\phi &+ a_{01} e_{01} + a_{02} e_{02} + a_{03} e_{03} \\ &+ a_{12} e_{12} + a_{13} e_{13} + a_{23} e_{23} \\ &+ a_\omega e_\omega \end{aligned} \tag{5}$$

where the sum is now over subsets σ of the set $\{0, 1, 2, 3\}$ whose size is 0, 2 or 4.

The *reverse* of an element A is obtained by reversing the order of the subscripts of the basis elements in equation (1). It is denoted by $\overline{A}$. The reverse of the product of two elements is the product of their reverses in the other order, $\overline{ab} = \overline{b}\,\overline{a}$. An element of grade 0, 1 or 4 is equal to its own reverse; the reverse of a bivector or trivector is the negative of itself.

An *inner product* (denoted by $\cdot$) and an *outer product* (denoted by $\wedge$) are defined in

$\mathcal{G}_4$ as follows.

$$a \cdot b = \tfrac{1}{2}(ab + ba) \tag{6}$$

$$a \wedge b = \tfrac{1}{2}(ab - ba) \tag{7}$$

These two products both satisfy the expected rules in terms of associativity and distribution. The inner product is commutative, and the outer product is anti-commutative. (Note that these products are defined here for any elements a and b in $\mathcal{G}_4$ (Mullineux 2002). This is different to the conventional approach of limiting the definition to vectors and then, for example, using an expression of the form $a \wedge b \wedge c$ to construct trivectors.)

The algebra $\mathcal{G}_4$ is a model of three dimensional projective space. The typical vector

$$p \;=\; We_0 + Xe_1 + Ye_2 + Ze_3$$

corresponds to the cartesian point $(X/W, Y/W, Z/W)$ assuming that W is non-zero (and it corresponds to a point at infinity if it is).

Further if p and q are two vectors, then $p \wedge q$ represents the (infinite) line on which the corresponding points lie (Mullineux and Simpson 2011, Dorst *et al.* 2007, González Calvet 2007). If r is a third vector, then, given the above definitions of the inner and outer products, the point corresponding to r lies on the line if and only if $(p \wedge q) \cdot r = 0$.

3. Transforms

Suppose that $S \in \mathcal{G}_4$ is an element of even grade. Suppose also that $p \in \mathcal{G}_4$ is a vector and so represents a point in (projective) 3-space. The product $\overline{S}pS$ has odd grade and is equal to its own reverse, and so it is also a vector (Cripps and Mullineux 2012). Hence the mapping

$$F_S : p \;\mapsto\; \overline{S}\, p\, S \tag{8}$$

defines a transform of the points of 3-space. If S has the property that $\overline{S}S = 1$, then the coefficient of e_0 in $F_S(p)$ is the same as that in p. However, since points are represented projectively, there is no need to make this assumption about S. In particular, if λ is a non-zero scalar, then S and λS generate the same transform.

It can be shown (Mullineux and Simpson 2011, Mullineux 2004) that this transform preserves lengths and angles and hence it is an isometry. Given a body in 3-space consisting of a (possibly infinite) number of points, then this can be transformed without distortion and hence the map F_S is a rigid-body transform. If S is a pseudo-scalar, then the map F_S is the identity. The pseudo-scalars form a subspace of dimension 2 of the space of even-grade elements which has dimension 8. There are 6 degrees of freedom in forming a rigid-body transform and hence the even-grade elements generate all of these transforms.

If S is a smooth function of a parameter t, then the body can be repositioned for each value of the parameter. Each result is called a *pose* of the body. Further as t varies, the body moves and $S = S(t)$ defines a *motion*. Any point p in the body correspondingly moves with the body and in doing so follows a curve in space.

Figure 1 shows an example motion of a body which is an L-shaped block. The curve shown is the curve traced out by one of the vertices. A smooth motion can be interpolated

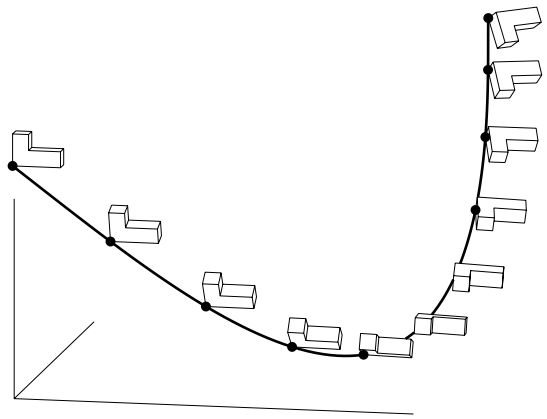

Figure 1. Example motion of an L-shaped block with the curve followed by one vertex

between the two poses in a number of ways. The form of interpolation used here is an extension of *slerp* (spherical linear interpolation) used with quaternions (Leeney 2009, Shoemake 1985). The extension requires the ability to raise an even-grade element of $\mathcal{G}_4$ to a non-integer power. The exponential and logarithm functions can be defined on even-grade elements of $\mathcal{G}_4$, both resulting in even-grade elements (Simpson and Mullineux 2009, Dorst *et al.* 2007), So, for an even-grade element $A \in \mathcal{G}_4$ and a general real power t, exponentiation can be defined by

$$A^t \;=\; \exp(t \log A)$$

Suppose two poses for a body are represented by even-grade elements S_0, $S_1 \in \mathcal{G}_4$. The slerp motion is generated by

$$S(t) \;=\; S_0 \, (\, \overline{S_0} \, S_1 \,)^t \tag{9}$$

for $0 \leq t \leq 1$. When $t = 0$, this is the same as S_0. Since $S_0\overline{S_0}$ is a pseudo-scalar, when $t = 1$, the pose generated is the same as that for S_1. The element $U = \overline{S_0}S_1$ has even grade and equation (9) can be rewritten as follows.

$$S(t) \;=\; S_0 \, U^t \tag{10}$$

Thus the motion can be regarded as the composition of the transform S_0 and a slerp interpolation between poses 1 and U.

Such motions interpolate two given poses. They are here referred to as motions of *order* 2. More generally a motion passing through n given poses is said to have *order* n.

The basic slerp construction can be extended to handle more than two control poses. It is less clear what the "degree" of the motion means and the order of the motion is simply interpreted as being the number of control poses.

4. Interpolation between tool poses

The conventional way for specifying the motion of a cutting tool is in terms of a sequence of cutter locations. These are passed to the controller of the machine tool which then performs curve interpolation (CI) to create the tool path. The algebra $\mathcal{G}_4$ is capable of

generating motion between two poses and the purpose of this section is to illustrate that it can create the standard forms of motion interpolation (MI) required for tool paths. Cases 4.1, 4.2, 4.3 and 4.4 are all simple interpolations between pairs of poses and are particular instances of the general such interpolation, case 4.6, which results in a screw motion (around the outside of a cylinder). Case 4.5 is a motion generated by composing two simple interpolations. The cutter locations are taken as poses for the tool. This means that the cartesian position is handled as required for 3-axis machining, and the orientation of the tool (relative to the workpiece) is catered for as needed for 4- and 5-axis applications.

4.1 *Linear interpolation*

Any translation can be generated using an even-grade element. This is achieved as follows. Suppose that $u = u_1e_1 + u_2e_2 + u_3e_3$ is a vector in the direction of the required motion, and that it is a unit vector in the sense that it has grade 1 and $\sqrt{[u_1^2 + u_2^2 + u_3^2]} = 1$. Then the even-grade element

$$T \;=\; 1 \;+\; \tfrac{1}{2}\varepsilon d e_0 u \tag{11}$$

creates a translation in the direction of u through distance $d \in \mathrm{R}$. As a partial check of this, consider the image of the point e_0 at the origin. This is given as follows.

$$\begin{aligned}
&\overline{T}e_0T \\
&= (1 - \tfrac{1}{2}\varepsilon d e_0 u)e_0(1 + \tfrac{1}{2}\varepsilon d e_0 u) \\
&= e_0 + \tfrac{1}{2}\varepsilon d e_0^2 u - \tfrac{1}{2}\varepsilon d e_0 u e_0 - \tfrac{1}{4}\varepsilon^2 d^2 e_0 u e_0 e_0 u \\
&= e_0 + \tfrac{1}{2}\varepsilon d e_0^2 u + \tfrac{1}{2}\varepsilon d e_0^2 u - \tfrac{1}{4}\varepsilon^2 d^2 e_0^3 u^2 \\
&= e_0 + \tfrac{1}{2}du + \tfrac{1}{2}du - \tfrac{1}{4}\varepsilon d^2 e_0 u^2 \\
&= (1 - \tfrac{1}{4}\varepsilon d^2 u^2)e_0 + du \\
&= e_0 + du
\end{aligned}$$

Figure 2 shows an example of MI between the poses $S_0 = 1$ and $S_1 = 1 + \varepsilon(4e_{01} + 2e_{02} + 3e_{03})$. The slerp form given by equation (9) is used and the motion is from the original position of the tool (at the origin) through a linear movement of 8 units in the x-direction, 4 in y, and 6 in z (these distances are twice the coefficients reflecting the factor of one half in equation (11)). The thick line shows the path of the centre of the base of the main cylinder.

4.2 *Circular interpolation*

A circular motion can be achieved in which the tool body remains vertical. This requires a rotation about a vertical axis which does not necessarily pass through the origin. It can be constructed as the composition of a translation taking the axis to the origin, a rotation about the new axis, and a translation back to the original axis. This results in the following even-grade element

$$R \;=\; (\cos\tfrac{1}{2}\theta) \;+\; (\sin\tfrac{1}{2}\theta)[\varepsilon y_c e_{01} - \varepsilon x_c e_{02} + e_{12}]$$

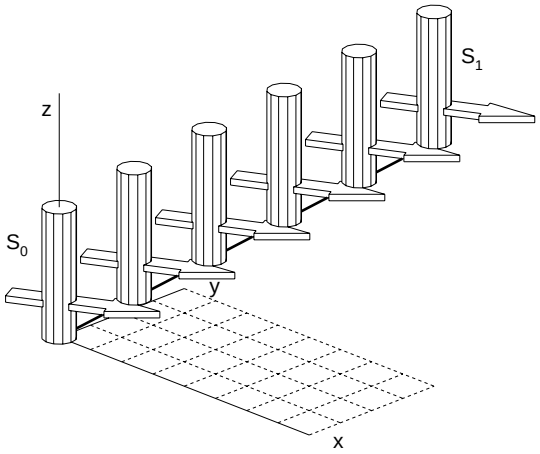

Figure 2. Linear MI (translation) between two given tool poses

where the point (x_c, y_c) represents the centre of the circular arc in the xy-plane and θ is the angle of rotation.

Figure 3 shows an example of such circular interpolation between the given end-poses. The angle θ is 150 degrees, and the centre of the rotation in the xy-plane is $(0, 4)$. This makes the even-grade element the following.

$$R \;=\; 0.2588 \;+\; 3.8637\varepsilon e_{01} \;+\; 0.9659 e_{12} \tag{12}$$

In the figure, the start of the motion is the pose $S_0 = 1$, and the end is the pose $S_1 = R$.

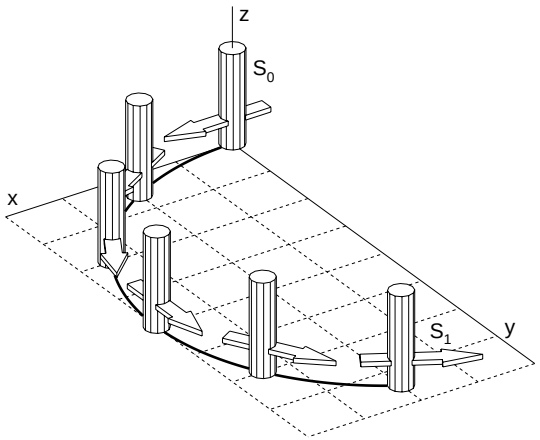

Figure 3. Circular MI between two given tool poses

4.3 *Free-form interpolation*

More general MI between two given poses S_0 and S_1 can be achieved with a construction based on the de Casteljau algorithm for free-form curves (Farin 2002). For example, a slerp motion, $S(t)$, of order 4 can be defined by the tableau shown in table 1 in which S_0 and S_1 are the given poses and Q_0 and Q_1 are additional intermediate control poses. As

$$
\begin{array}{lll}
S_0 & & \\
 & S_{01}(t) = S_0[\overline{S_0}Q_0]^t & \\
Q_0 & S_{02}(t) = S_{01}(t)[\overline{S_{01}(t)}S_{12}(t)]^t & \\
 & S_{12}(t) = Q_0[\overline{Q_0}Q_1]^t & S(t) = S_{02}(t)[\overline{S_{02}(t)}S_{13}(t)]^t \\
Q_1 & S_{13}(t) = S_{12}(t)[\overline{S_{12}(t)}S_{23}(t)]^t & \\
 & S_{23}(t) = Q_1[\overline{Q_1}S_1]^t & \\
S_1 & &
\end{array}
$$

Table 1. Tableau of poses for the de Casteljau algorithm

before, the parameter t goes from 0 to 1. Figure 4 provides an example of such a free-form MI using the control poses given in table 2. The arrow on the moving body (in its own local coordinate frame) points in the x direction. The first given pose S_0 represents a rotation about the z-axis through 90 degrees, and the first additional control pose Q_0 is this transform composed with a translation of 4 units in the y-direction. The second given pose S_1 is a translation of the original body through 8 units in the x-direction, and the second additional control pose Q_1 is a translation in the same direction through 4 units. The two additional control poses are shown with dashed lines in the figure.

S_0	$(1 + e_{12})/\sqrt{2}$
Q_0	$(1 + 2\varepsilon e_{01} + 2\varepsilon e_{02} + e_{12})/\sqrt{2}$
Q_1	$1 + 2\varepsilon e_{01}$
S_1	$1 + 4\varepsilon e_{01}$

Table 2. Control poses for the slerp motion of order 4 shown in figure 4

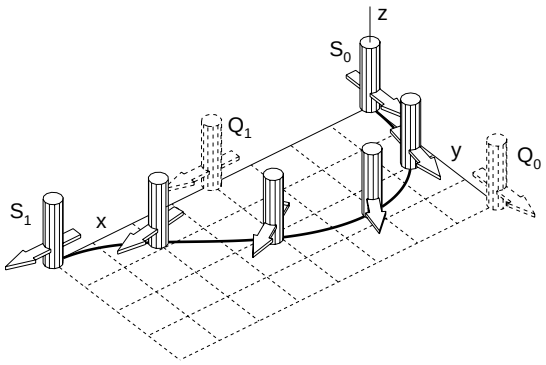

Figure 4. Free-form MI between two given tool poses with two additional control poses (shown dashed) as given in table 2

4.4 *Linear interpolation with twist*

The previous examples of MI between poses are all essentially motions in two dimensions. Truly three dimensional motions can also be achieved. One way to construct these is

using a composition of appropriate individual two-dimensional motions. Figure 5 shows a translation along the x-axis through 8 units given by

$$T \;=\; 1 \,+\, 4\varepsilon e_{01}$$

composed with a rotation through -60 degrees about the (positive) x-axis given by

$$R \;=\; c \,-\, se_{23}$$

where $c = \cos\frac{\pi}{6}$ and $s = \sin\frac{\pi}{6}$.

The slerp motion shown uses equation (9) with the tool moving from its original pose $S_0 = 1$ to the final pose

$$\begin{aligned} S_1 = RT \;&=\; TR \\ &= c + 4\varepsilon c e_{01} - s e_{23} - 4\varepsilon s\omega \\ &= 0.8660 + 3.4641\varepsilon e_{01} - 0.5 e_{23} - 2\varepsilon\omega \end{aligned}$$

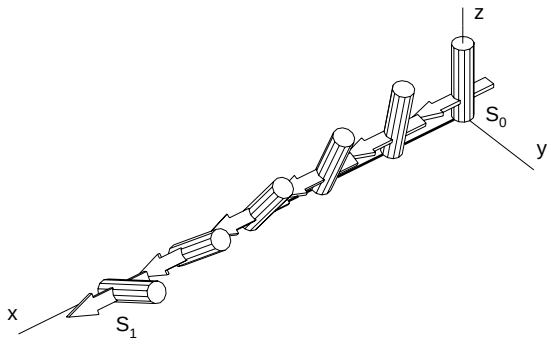

Figure 5. Linear MI with twist between two given tool poses

4.5 *Circular interpolation with twist*

The above composition of a translation and rotation gives the "expected" result, which is that the tool tip translates and the tool rotates about its line of motion. This is because the axis of the rotation is in the direction of translation, and the rotation and translation commute. Care is required for other compositions. For example, consider the case of composing the circular interpolation given in figure 3 with a rotation through 45 degrees about the (local) x-axis of the tool. The former rotation is given as R in equation (12) and the latter is given by the following even-grade element.

$$Q \;=\; (\cos\tfrac{\pi}{8}) + (\sin\tfrac{\pi}{8})e_{23}$$

A MI is required between the initial pose $S_0 = 1$ and the final pose $S_1 = QR$. This can be achieved using equation (9). However this does not create a motion lying within the xy-plane. Such a motion can be obtained by composing the two individual motions. The

motion between 1 and Q is simply Q^t, and that between 1 and R is R^t. Their composition gives the following motion

$$S(t) \;=\; [Q^t][R^t] \qquad\qquad 0 \le t \le 1$$

between the poses

$$\begin{aligned} S_0 &= 1 \\ S_1 &= QR \\ &= 0.239 + 3.5706\varepsilon e_{01} + 0.892 e_{12} - 0.370 e_{13} \\ &\quad +0.099 e_{23} + 1.479\varepsilon\omega \end{aligned}$$

as shown in figure 6.

More generally, if $S(t) = S_0(\overline{S_0}S_1)^t$ is a motion between poses S_0 and S_1, and Q is a motion to be applied to the moving shape (in its own local coordinate frame), then the composed motion takes the form

$$[Q^t][S_0(\overline{S_0}S_1)^t]$$

and gives a motion in which the tool tip moves along the arc and the tool rotates about the local tangent to the arc.

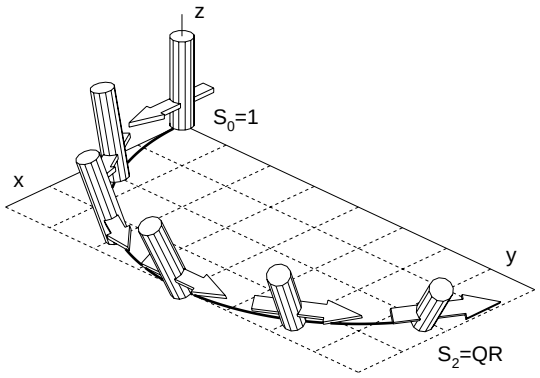

Figure 6. Circular MI with twist between two given tool poses

4.6 *General interpolation*

Suppose that $u = u_1e_1 + u_2e_2 + u_3e_3$ is a unit vector representing a direction. Then equation (11) gives the element T corresponding to a general translation in this direction through distance d. Set $b = ue_{123}$ which can be regarded as the corresponding bivector for the direction. Then the line through a point p in the given direction is represented by the following even-grade element (Mullineux 2002, Mullineux and Simpson 2011).

$$\ell \;=\; -b \;-\; \varepsilon e_0(b \wedge p) \tag{13}$$

Further, a rotation about this line (as the axis) through angle θ is generated by the following even-grade element (Mullineux and Simpson 2011).

$$\begin{aligned} R &= (\cos\tfrac{1}{2}\theta) \ + \ (\sin\tfrac{1}{2}\theta)[b + \varepsilon e_0(b \wedge p)] \\ &= (\cos\tfrac{1}{2}\theta) \ - \ \varepsilon(\sin\tfrac{1}{2}\theta)\ell\omega \end{aligned} \tag{14}$$

A particular case is when $p = e_0$ so that the axis passes through the origin. Since u has unit length, it follows that $b^2 = -1$, and R is given by

$$R \ = \ (\cos\tfrac{1}{2}\theta) \ + \ (\sin\tfrac{1}{2}\theta)b \ = \ \exp(\tfrac{1}{2}\theta b) \tag{15}$$

which is the equivalent of Euler's formula in the theory of complex numbers.

Clearly a transform which is a composition of translations and rotations can be obtained by multiplying the corresponding even-grade elements given by equations (11) and (14). Conversely, Chasles's theorem (e.g. (Belta and Kumar 2002, Nikravesh 1998)) says that any rigid-body transform can be expressed as the composition of a rotation about an axis and a translation along that axis. Hence, as noted before, the set of even-grade elements is precisely the set of rigid-body transforms.

Chasles's theorem says that a *screw* motion is generated. This is the most general form of interpolation generated by an even-grade element of $\mathcal{G}_4$. Further, if R and T represent the relevant rotation and translation, then, because they share the same axis, it follows that they commute, $TR = RT$. An example of a screw motion is shown in figure 7. The axis is the line $x = 0$, $z = -1$ in cartesian coordinates. In the above notation, $p = e_0 - e_3$ is a point on the line, and $b = -e_{13}$ is the bivector for its direction. Hence, from equation (13), the axis is the following.

$$\ell \ = \ -\varepsilon e_{01} + e_{13}$$

The body starts at the origin with the identity transform ($S_0 = 1$). At the end of the motion, at the right in the figure, the transform is generated by the even-grade element $S_1 = U = RT = TR$, where the rotation R is through 90 degrees and the translation T is over 2 units. These and their product U (ignoring higher powers of ε) are as follows.

$$\begin{aligned} R &= [1 + \varepsilon e_{01} - e_{13}]/\sqrt{2} \\ T &= 1 + \varepsilon e_{02} \\ U &= [1 + \varepsilon e_{01} + \varepsilon e_{02} - e_{13} + \varepsilon\omega] \end{aligned}$$

The motion itself is given by the variable element $S(t) = U^t$, $0 \le t \le 1$. As seen in figure 7, the motion is around a cylinder along the axis of the screw with the moving body remaining at a constant angle to the surface.

Elements T and R can be rewritten in exponential form giving the following.

$$T = 1 \; - \; \tfrac{1}{2}\varepsilon db\omega \;\; = \;\; \exp(-\tfrac{1}{2}\varepsilon db\omega) \tag{16}$$

$$\begin{aligned} R &= (\cos\tfrac{1}{2}\theta) \; - \; \varepsilon(\sin\tfrac{1}{2}\theta)\ell\omega \\ &= \exp(-\tfrac{1}{2}\varepsilon\theta\ell\omega) \end{aligned} \tag{17}$$

$$\begin{aligned} U &= RT \; = \; TR \\ &= \exp[-\tfrac{1}{2}\varepsilon(\theta\ell + db)\omega] \end{aligned} \tag{18}$$

Consideration of the exponential form of U^t shows that during the motion the angle of rotation and the distance of translation both vary linearly. Hence the path of any point in the body is a helix. Figure 7 also shows the path of the centre of the base of the tool: this is a helix on the surface of the cylinder. In view of equation (10), it follows that a

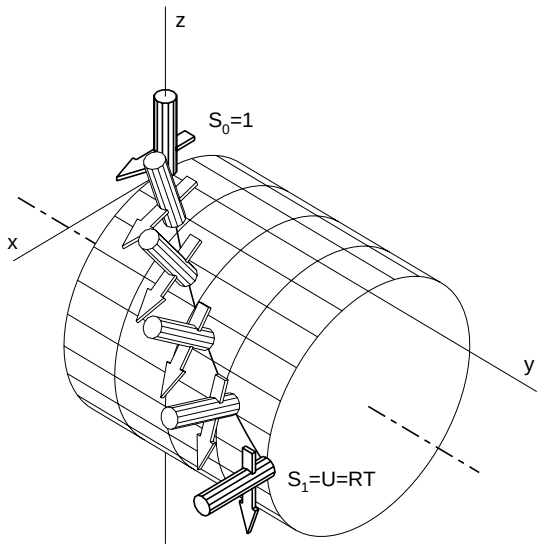

Figure 7. Typical screw motion as a slerp interpolation between $S_0 = 1$ and $S_1 = U$ using equation (9): $S(t) = U^t$ for $0 \leq t \leq 1$

slerp interpolation between poses S_0 and S_1 is a screw transform which is determined by $U = \overline{S_0}S_1$. Hence, the path of each point in the body is a helix and so is a non-rational curve (Farin 2002).

5. Degrees of freedom and direction

The previous section shows how to generate MI between two given poses of a tool. The interest is now in using this idea to generate the path of a tool along a sequence of given poses in which any two adjacent poses are roughly similar so that there is no sudden change in the implied overall motion. A pose of a general body has six degrees of freedom. If the body represents the cutter of a machine tool, then it can be treated as a cylinder and so one of the degrees of freedom, the rotation about its axis, has no real effect.

However, that sixth degree of freedom is significant when forming motions between two tool poses. This is illustrated in figure 8. Here the body is shown as a cylinder together

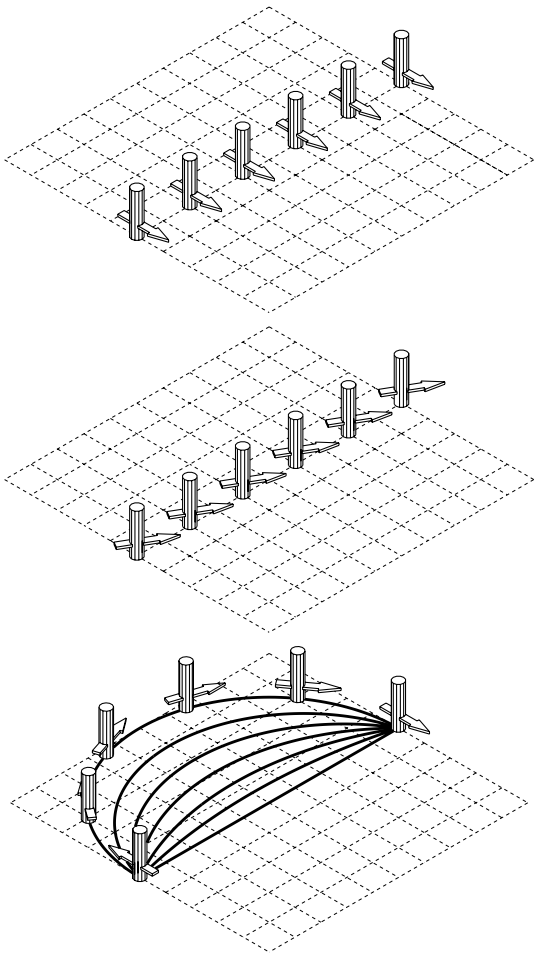

Figure 8. Three examples of linear MI – top: no rotation about vertical axis, middle: equal rotations, bottom: six cases of different rotations.

with an arrow to indicate the rotation about the axis. In the first part of the figure, the pose on the left is the body in its original state. On the right is the image of the body after a pure translation. The result of interpolating between these is shown as a translation in a straight line. In the second part of the figure, the same rotation about the axis is imposed on both the given poses. Again a motion is obtained which is a straight line translation. In the third part of figure 8, different rotations are applied to the end-poses. The paths followed are now the circular arcs shown; the motion is illustrated in the case when the difference in the rotations is 180 degrees. It is seen that the form of the motion depends upon the relative rotations of the end-poses.

This means that if the interpolation technique is to be used to simulate the motion of a cutting tool between two given tool poses, then it is important to ensure that the relative rotation between the end-poses is consistent with the desired direction of motion of the tool at those positions.

The second issue for consideration is the direction of the screw motion between two given poses. If these are S_0 and S_1, then, as in equation (9), the motion is $S(t) = S_0 U^t$ where $U = \overline{S_0} S_1$. If S_1 is replaced by λS_1, where λ is a non-zero scalar, then the position of any point in the body in the final pose is unchanged, since projective space is being used. This also applies to any intermediate pose along the motion provided $\lambda > 0$.

However, if the scalar is negative, the motion is still a screw about the same axis but with the rotation going in the opposite direction.

This can be seen in the particular case of a rotation about an axis through the origin in the direction of the unit bivector b. As in equation (15) this is given by the following even-grade element.

$$R \;=\; (\cos\tfrac{1}{2}\theta) + (\sin\tfrac{1}{2}\theta)b \;=\; \exp(\tfrac{1}{2}\theta b)$$

The negative of this is then

$$\begin{aligned} -R &= -\exp[\tfrac{1}{2}\theta b] \\ &= \exp[\pi b]\ \exp[\tfrac{1}{2}\theta b] \\ &= \exp[\tfrac{1}{2}(2\pi-\theta)(-b)] \end{aligned}$$

which represents a rotation through angle $(2\pi-\theta)$ about the axis $-b$ going in the opposite direction. Hence $-R$ is the same rotation as R. However, since the axis is reversed the rotation is now in the opposite direction. In particular, $(-R)^t = \exp[\frac{1}{2}t(2\pi-\theta)(-b)])$ represents a rotation through angle $t(2\pi-\theta)$ turning one way, while $R^t = \exp[\frac{1}{2}t\theta b])$ represents a rotation through angle $t\theta$ going the other way.

This idea extends to the general rotation R given by equation (17), and hence to the general transform U given by equation (18).

Figure 9 shows an example of a screw motion and its negative: the upper right is an isometric view, the others are views along the main axes. Both motions involve the same translation (in the y-direction), but one rotation is through 90 degrees and the other through 270 degrees. When interpolating between tool poses, it is assumed that the poses

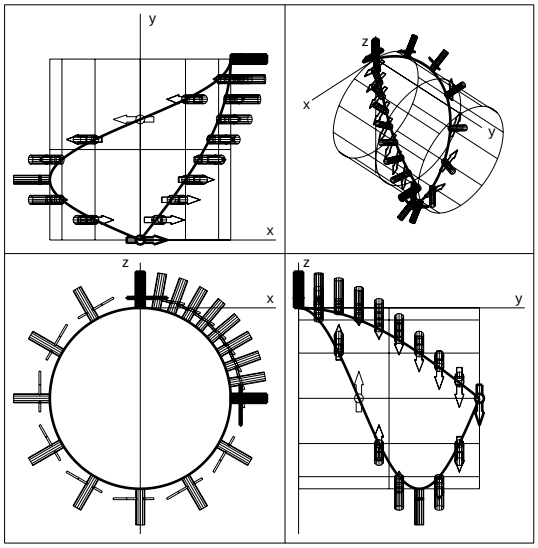

Figure 9. Example of positive and negative screw motions

S_0 and S_1 are similar in translation and rotation. So the screw motion $U = \overline{S_0}S_1$ needs to be through an angle close to zero. Equation (18) shows how to write U as a product of a translation and a rotation as given by equations (11) and (14) respectively. The scalar part (that is the coefficient of e_ϕ) in this product is $\cos\frac{1}{2}\theta$. So, when interpolating between

S_0 and S_1, a check is made on the sign of the scalar part of U. If this is negative, then the sign of S_1 (and hence of U) is reversed to obtain the screw motion in the appropriate direction.

6. Comparison examples

When a free-form surface is manufactured by milling, the machine tool is driven over the surface. It is controlled by the controller of the machine tool which itself receives instructions based upon a CAD (geometric) model of the surface. Those instructions are essentially "move to" commands obtained by sampling at discrete precision points on the surface. It is assumed here that the controller essentially performs "straight line" movements between the precision points. It is thus producing a piece-wise linear curve in space.

The purpose of this section is to compare CI and MI. Sequences of precision points or precision poses are obtained by sampling a surface. A piece-wise curve or motion is then created for each sequence. Each "piece" is based on either a pair or triple of consecutive precision points or poses from the sequence. For a pair, the piece is formed by interpolating linearly with the two items forming the ends of the piece; for a triple, the interpolation is quadratic with the piece passing between the first and the last items of the triple.

Four cases are compared. These are linear and quadratic forms of CI and MI. The following subsections give more details.

Linear CI

Given two precision points $\mathbf{r}_0$ and $\mathbf{r}_1$, the simplest curve between them is a straight line segment of the form

$$(1-t)\mathbf{r}_0 \;+\; t\mathbf{r}_1 \qquad \text{for} \quad 0 \le t \le 1$$

Linear MI

Given two precision poses S_0 and S_1, the motion used between them is the basic slerp form given by equation (9)

$$S(t) \;=\; S_0\,(\overline{S_0}S_1)^t$$

Quadratic CI

If $\mathbf{r}_0$, $\mathbf{r}_1$, $\mathbf{r}_2$ are three consecutive precision points in (cartesian) space, then the following curve "piece"

$$(1-t)^2\mathbf{r}_0 \;+\; 2t(1-t)\mathbf{q} \;+\; t^2\mathbf{r}_2$$

where

$$\mathbf{q} \;=\; -\tfrac{1}{2}\mathbf{r}_0 + 2\mathbf{r}_1 - \tfrac{1}{2}\mathbf{r}_2$$

represents a curve segment (for $0 \leq t \leq 1$) passing between $\mathbf{r}_0$ and $\mathbf{r}_2$ and passing through $\mathbf{r}_1$ when $t = \frac{1}{2}$.

Quadratic MI

Suppose a triple of control poses is given: S_0, S_1, S_2. As with the curve case, the following additive combination

$$S(t) \;=\; (1-t)^2 S_0 \;+\; 2t(1-t)Q_0 \;+\; t^2 S_2$$

where

$$Q_0 \;=\; -\tfrac{1}{2}S_0 \;+\; 2S_1 \;-\; \tfrac{1}{2}S_2 \tag{19}$$

gives a motion (of order 3) through these poses.

However the interest here is in creating the motion "piece" using the slerp construction. Given the two end-poses S_0 and S_2, and a third control pose Q, a motion of order 3 can be constructed, with multiplicative combinations, using the de Casteljau algorithm as follows.

$$S(t) \;=\; S_{01}(t)\,[\overline{S_{01}(t)}S_{12}(t)]^t$$

where

$$S_{01}(t) = S_0\,(\overline{S_0}Q)^t \quad \text{and} \quad S_{12}(t) = Q\,(\overline{Q}S_2)^t$$

This motion certainly starts at $S(0) = S_0$ and finishes at $S(1) = S_2$. The middle control pose, Q, needs to be chosen to ensure that

$$S_1 \;=\; S(\tfrac{1}{2}) \;=\; S_0\,(\overline{S_0}Q)^{\frac{1}{2}}\,[(\overline{Q}S_0)^{\frac{1}{2}}\overline{S_0}Q(\overline{Q}S_2)^{\frac{1}{2}}]^{\frac{1}{2}} \;=\; S_0\,(\overline{S_0}Q)^{\frac{1}{2}}\,[(\overline{S_0}Q)^{\frac{1}{2}}\,(\overline{Q}S_2)^{\frac{1}{2}}]^{\frac{1}{2}} \tag{20}$$

The fact that multiplication is not commutative means that the brackets here cannot be removed to simplify the expression. Instead, a numerical technique is needed to solve the non-linear equation. An iterative search works successfully and experience has shown that a good starting point for the iterations is the value of Q_0 given by equation (19). In fact, it is also found that with the examples given here, the value of Q_0 is itself good enough as the solution to equation (20) with small error. This is partly because the poses S_0 and S_2 are close together (particularly in the case when $n = 10$ meaning that more steps are taken, see below).

For the comparisons, two free-form surfaces are used which are Bézier patches of the form $\mathbf{r}(u, v)$ with parameters u and v. The use of the Bézier form is purely for convenience: the approach can be applied to any surface representation for which evaluation of points and first derivatives is available.

A regular grid of precision points or poses is taken over each patch. It is between these that interpolation to obtain the "pieces" of the piece-wise curve or motion is carried out. The grid depends upon a given number n. Isoparametric lines are considered where u and v take the values (i/n) for $0 \leq i \leq n$. There are $(n+1)$ such lines in each direction. Figure 10 shows an example of the precision points and poses at the intersections of these lines in the case when $n = 5$.

This regular grid forms the end-points or end-poses for each "piece". For the quadratic cases, the point or pose halfway (in parametric terms) between each pair from the grid is also used giving the middle point or pose in a triple of consecutive precision points or poses. For the CI cases, the precision points are simply the points on the surface. For

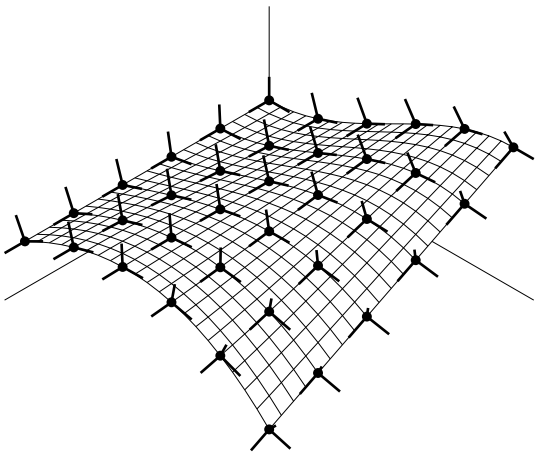

Figure 10. Grid of precision points (dots) and poses (reference frames) (for $n = 5$) on a Bézier surface

the MI cases, the precision poses are obtained as follows. The tool (body) is represented by a cylinder with its origin in the centre of its base and the local z-direction along its axis. The x- and y-axes lie in the base of the tool (with the x-axis being in the direction of the arrow shown in previous figures). At any precision pose on the surface, the tool is oriented so that its z-axis is in the direction of the local surface normal (as would be necessary if machining with a flat-bottomed cutter). The unit surface normal is obtained by normalising $\mathbf{r}_u \times \mathbf{r}_v$, where $\mathbf{r}_u$ and $\mathbf{r}_v$ are the partial derivatives with respect to the parameters. The x-axis is then aligned with the required direction of motion which for the purpose of the examples here is taken as being as the appropriate one of $\mathbf{r}_u$ and $\mathbf{r}_v$.

To estimate the error between the interpolation and the true surface, each piece (of the piece-wise curve or motion) is itself divided into n parts, equally spaced (parametrically), giving $(n + 1)$ points along it, including the ends; for the case of motion, the point used is the origin of the local frame. Each point represents the centre of the base of the tool as it moves. The distance of each point from the surface is evaluated. This is done by using a direct search method, varying the u and v, to find the nearest point on the surface. The end-points of each "piece" naturally lie on the surface. The largest distance is taken as a measure of the error in the interpolation. A sign is given to the error depending on which side of the surface the tool point lies: the direction of the surface normal is taken as positive.

The results for two example surfaces are presented in the following subsections. The first surface is a biquadratic patch, giving an interpolation between a circle and a straight line segment. The second example is bicubic and involves points of inflexion.

For each example its control points are given together with a figure showing its form. Two values of n are used, 5 and 10. Tables are presented giving the largest errors found in the positive and negative directions for the two values of n. Also given is the sum of these two extremes which represents the range of the errors. The error results are also illustrated by plots showing the errors along the lines (in parametric space) of each of the grids. In the case of $n = 5$, a scaling factor of 10 is applied; the factor is 100 when $n = 10$.

The scaling means that the error plots for the basic piece-wise linear CI cases cannot be easily interpreted; they are provided simply for comparison and the illustration that in the other cases the errors are considerably reduced. Following the examples, a summary of the results is provided.

6.1 *Rational Bézier biquadratic surface*

This example is a rational Bézier biquadratic patch. It is a blend between linear and circular boundaries. The homogeneous coordinates of its nine control points are shown in table 3 and the surface is plotted in figure 11. The extreme error values are listed in table 4 and figure 12 gives the error plots.

X	Y	Z	W	X	Y	Z	W	X	Y	Z	W
0.0	0.0	0.0	1.0	0.0	0.0	2.0	0.0	4.0	0.0	0.0	1.0
0.0	2.0	0.0	1.0	0.0	0.0	2.0	0.0	4.0	2.0	0.0	1.0
0.0	4.0	0.0	1.0	2.0	4.0	0.0	1.0	4.0	4.0	0.0	1.0

Table 3. Homogeneous control points for rational Bézier biquadratic patch

	n	Max – error	Max + error	Error range
linear CI	5	0.147447	0.000000	0.147447
linear MI	5	0.001229	0.001974	0.003203
quadratic CI	5	0.006902	0.005086	0.011989
quadratic MI	5	0.000939	0.000704	0.001643
linear CI	10	0.038839	0.000000	0.038839
linear MI	10	0.000269	0.000348	0.000617
quadratic CI	10	0.000826	0.000732	0.001558
quadratic MI	10	0.000166	0.000145	0.000311

Table 4. Error values for Bézier biquadratic patch

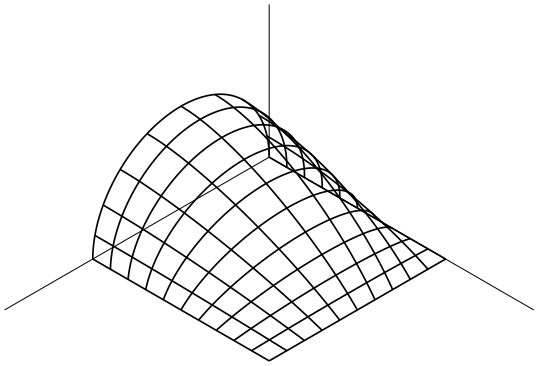

Figure 11. Rational Bézier biquadratic patch

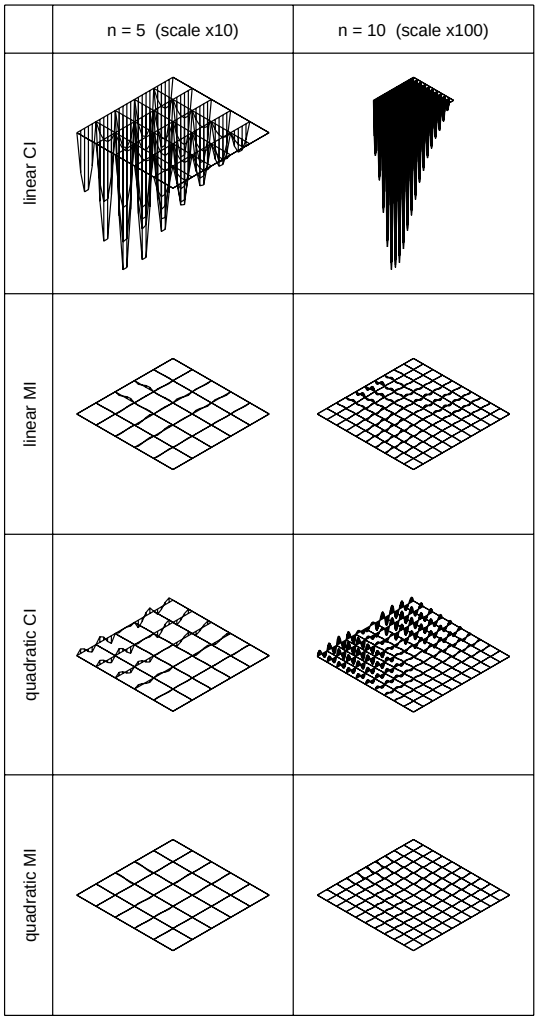

Figure 12. Error plots for rational Bézier biquadratic patch

6.2 *Bézier bicubic surface*

This example is a non-rational patch. It is bicubic and its 16 control points appear in table 5. The surface itself is shown in figure 13. The maximum positive and negative errors are given in table 6 and the error plots are seen in figure 14.

x	y	z	x	y	z	x	y	z	x	y	z
0.0	0.0	1.0	2.0	0.0	1.0	4.0	0.0	1.0	6.0	0.0	1.0
0.0	2.0	1.0	2.0	2.0	1.0	4.0	2.0	2.0	6.0	2.0	2.0
0.0	4.0	3.0	2.0	4.0	3.0	4.0	4.0	2.0	6.0	4.0	2.0
0.0	6.0	3.0	2.0	6.0	2.0	4.0	6.0	1.0	6.0	6.0	0.0

Table 5. Cartesian control points for Bézier bicubic patch

	n	Max – error	Max + error	Error range
linear CI	5	0.050771	0.046342	0.097113
linear MI	5	0.001993	0.001992	0.003986
quadratic CI	5	0.001483	0.001531	0.003015
quadratic MI	5	0.000994	0.000953	0.001947
linear CI	10	0.014178	0.013439	0.027617
linear MI	10	0.000256	0.000246	0.000502
quadratic CI	10	0.000196	0.000198	0.000394
quadratic MI	10	0.000124	0.000126	0.000250

Table 6. Error values for Bézier bicubic patch

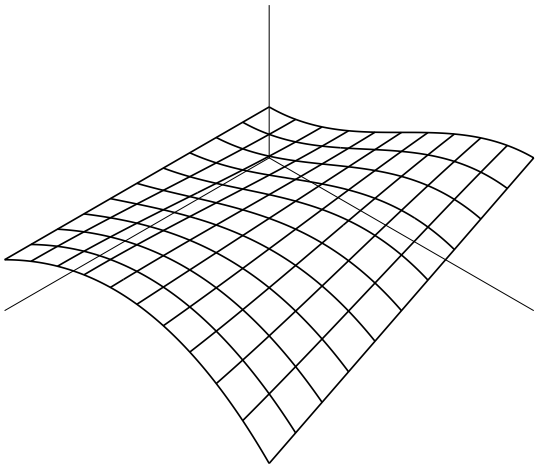

Figure 13. Bézier bicubic patch

6.3 *Summary of examples*

A number of observations relating to the examples can be made. In each example, the errors are reduced by making the grid finer (that is going from $n = 5$ to $n = 10$). This is natural since the interval over which interpolation takes place is reduced.

More significant is the fact that the error is reduced by passing from a piece-wise curve to a piece-wise motion. In a sense this is because the order of the curve forming the path of any point in the body is increased. As noted previously, the concept of "degree" does not really apply to the curves generated by the slerp construction used.

So, when linear interpolation is used (which is of course straightforward to implement), the motion approach is substantially better than that based on curves. Indeed, linear MI compares well with quadratic CI. The errors for these two cases are all similar in size.

7. Conclusions

It has been seen that the form of geometric algebra, $\mathcal{G}_4$, is capable of modelling three dimensional geometry and of representing exactly rigid-body transforms of that geometry. The interest has been in investigating whether the algebra can be applied to represent the motions of a cutting tool. Motion interpolation (MI) between two tool poses has been shown to be possible using the idea of spherical linear interpolation (slerp). This creates a screw motion, with the path of any point (in the body being transformed) being an

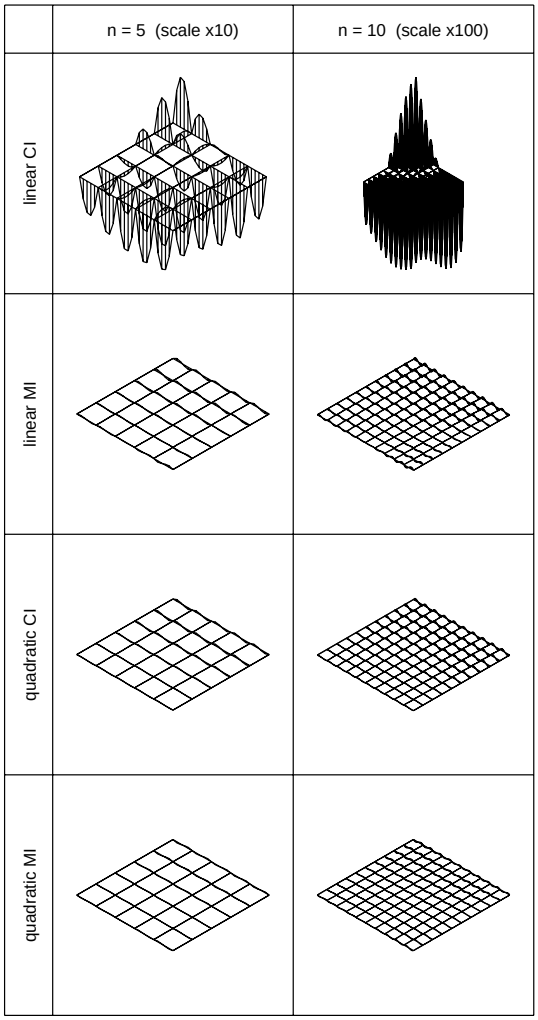

Figure 14. Error plots for Bézier bicubic patch

exact helix around a circular cylinder. This means that the approach is capable of representing precisely standard machining operations such as linear and circular interpolation. Furthermore, MIs can be composed to generate more complex forms.

MI has been compared with curve interpolation (CI) by considering interpolation between adjacent members of sequences of poses constructed over free-form surfaces. It was found that the interpolation error for the MI cases is smaller than for the CI cases (when the same form of interpolation is used). This is because the interpolated motion naturally changes the tool orientation and hence better follows the original surface. In particular, linear MI of poses compares well with quadratic CI of points.

Hence the use of precision poses (rather than precision points) seems a more natural way to transfer information to a machine tool controller. This suggests that there are advantages if such controllers were able to accept, manipulate and interpolate pose data.

Acknowledgement

The authors gratefully acknowledge the support of the Engineering and Physical Sciences Research Council (EPSRC) in funding a project entitled “Algebraic modelling of 5-axis

tool path motions" (ref: EP/L006316/1).

References

Ahlers, S. G. and McCarthy, J. M., 2001. The Clifford algebra and the optimization of robot design. In: *Geometric Algebra with Applications in Science and Engineering*, Bayro Corrochano, E., Sobczyk, G., eds., Birkhäuser, Boston, 235–251.

Akyar, B., 2008. Dual quaternions in spatial kinematics in an algebraic sense. *Turkish Journal of Mathematics*, 32(4), 373–391.

Annoni, M., Bardine, A., Campanelli, S., Foglia, P. and Prete, C. A., 2012. A real-time configurable NURBS interpolator with bounded acceleration, jerk and chord error. *Computer-Aided Design*, 44(6), 509–521.

Azmy, E. W., 2013. Exact solution of inverse kinematic problem of 6R serial manipulators using Clifford algebra. *Robotica*, 31(3), 417–422.

Bayro-Corrochano, E. and Zamora-Esquivel, J., 2006. Differential and inverse kinematics of robot devices using conformal geometric algebra". *Robotica*, 25(1), 44–61.

Belta, C. and Kumar, V., 2002. Euclidean metrics for motion generation on $SE(3)$. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 216(1), 47–60.

Beudaert, X., Pechard, P.-Y. and Tournier, C., 2011. 5-axis tool path smoothing based on drive constraints. *International Journal of Machine Tools & Manufacture*, 51(12), 958–965.

Bhuiya, M. S. H. and Tutunea-Fatan, O. R., 2013. Reduction of geometry-based errors in five-axis machining through enhanced 5D interpolation. *International Journal of Advanced Manufacturing Technology*, 64(1–4), 305–317.

Bottema, O. and Roth, B., 1979. *Theoretical Kinematics*, North-Holland, Amsterdam.

Cibura, C. and Dorst, L., 2011. Determining conformal transformations in R^n from minimal correspondence data. *Mathematical Methods in the Applied Sciences*, 34(16), 2031–2046.

Cripps, R. J. and Mullineux, G., 2012. Constructing 3D motions from curvature and torsion profiles. *Computer-Aided Design*, 44(5), 379–387.

Dell'Acqua, A., Sarti, A. and Turbaro, S., 2008. 3D motion from structures of points, lines and planes. *Image and Vision Computing*, 26(4), 529–549.

Dorst, L., 2010. "Tutorial: structure-preserving representation of Euclidean motions through conformal geometric algebra". In: *Geometric Algebra Computing*, Bayro-Corrochano E. and Scheuermann, G., eds., Springer-Verlag, London.

Dorst, L., Fontijne, D. and Mann, S., 2007. *Geometric Algebra for Computer Science: An Object-oriented Approach to Geometry*, Morgan Kaufmann, Amsterdam.

Etzel, K. R. and McCarthy, J. M., 1999. Interpolation of spatial displacements using the Clifford algebra of E^4. *Transactions of the ASME: Journal of Mechanical Design*, 121(1), 39–44.

Fang, Y. C., Hsieh, C. C., Kim, M. J., Chang, J. J. and Woo, T. C., 1998. Real time motion fairing with unit quaternions. *Computer-Aided Design*, 30(3), 191–198.

Farin, G., 2002. *Curves and Surfaces for CAGD: A Practical Guide*, 5th edition, Morgan Kaufmann, San Francisco.

González Calvet R., 2007. *Treatise of Plane Geometry through Geometric Algebra*, TIMSAC, Cerdanyola del Vallés.

Heng, M. and Erkorkmaz, K., 2010. Design of a NURBS interpolator with minimal feed fluctuation and continuous feed modulation capability. *International Journal of Machine Tools & Manufacture*, 50(3), 281–293.

Hofer, M., Pottmann, H. and Ravani, B., 2004. From curve design algorithms to the design of rigid body motions. *The Visual Computer*, 20(5), 279–297.

Jin, Z. and Ge, Q. J., 2010. Constrained motion interpolation for planar open kinematic chains. *Mechanism and Machine Theory*, 45(11), 1721–1732.

Koren, Y. and Lin, R.-S., 1995. "Five-axis surface interpolators", *CIRP Annals – Manufacturing Technology*, 44(1), 379–382.

Leeney, M., 2009. Fast quaternion slerp. *International Journal of Computer Mathematics*, 86(1), 79–84.

Lei, W. T. and Wang, S. B., 2009. Robust real-time NURBS path interpolators. *International Journal of Machine Tools & Manufacture*, 49(7–8), 625–633.

Li, J.-G., Qiu, M.-M., Zhang, T.-H. and Li, Z.-X., 2012. A practical real-time non-uniform rational B-spline curve interpolator for computer numerical control machining. *Proceeding of the Institution of Mechanical Engineers, Part C; Journal of Mechanical Engineering Science*, 226(C4), 1068–1083.

Li, P., Guo, R., Wang, P. and Yan Huang, Y., 2009. Research on tool path planning for five-axis machining. In: *Proceedings of IEEE International Conference on Industrial Engineering and Engineering Management (IEEM 2009)*, Hong Kong, 2338–2342.

Liang, H. and Li, X., 2013. "Five-axis STEP-NC controller for machining of surfaces". *The International Journal of Advanced Manufacturing Technology* , 68(9–12), 2791–2800.

Liu, Q., Jin, X. J. and Long, Y. H., 2010. A real-time high-precision interpolation algorithm for general-typed parametric curves in CNC machine tools. *International Journal of Computer Integrated Manufacturing*, 23(2), 168–176.

Mohan, S., Kweon, S.-H., Lee, D.-M. and Yang, S.-H., 2008. Parametric NURBS curve interpolators: a review. *International Journal of Precision Engineering and Manufacture*, 9(2), 84–92.

Mullineux, G., 2002. Clifford algebra of three dimensional geometry. *Robotica*, 20(6), 687–697.

Mullineux, G., 2004. Modeling spatial displacements using Clifford algebra. *Transactions of the ASME: Journal of Mechanical Design*, 126(3), 420–424.

Mullineux, G. and Simpson, L. C., 2011. Rigid-body transforms using symbolic infinitesimals. In: *Guide to Geometric Algebra in Practice*, Dorst, L. and Lasenby, J. (eds.), Springer, London, 353–369.

Nikravesh, P. E., 1998. *Computer-Aided Analysis of Mechanical Systems*, Prentice-Hall, London.

Perwass, C., 2009. *Geometric Algebra with Applications in Engineering*, Springer, Berlin.

Purwar, A. and Ge, Q. J., 2005. On the effect of dual weights in computer aided design of rational motions. *Transactions of the ASME: Journal of Mechanical Design*, 127(5), 967–972.

Purwar, A., Chi, X. and Ge, Q. J., 2008. Automatic fairing of two-parameter rational B-spline motion. *Transactions of the ASME: Journal of Mechanical Design*, 130(1), 011003:1–7.

Purwar, A., Jin, Z. and Ge, Q. J., 2008. Rational motion interpolation under kinematic constraints of spherical 6R closed chains. *Transactions of the ASME: Journal of Mechanical Design*, 130(6), 062301:1–9.

Rauch, M., Laguionie, R., Hascoet, J.-Y. and Suh, S.-H., 2012. An advanced STEP-NC controller for intelligent machining processes. *Robotics and Computer-Integrated Manufacturing*, 28(3), 375–384.

Röschel, O., 1998. Rational motion design – a survey. *Computer-Aided Design*, 30(3), 169–178.

Sariyildiz, E. and Temeltas, H., 2012. A new formulation method for solving kinematic problems of multiarm robot systems using quaternion algebra in the screw theory framework. *Turkish Journal of Electrical Engineering and Computer Science*, 20(4), 607–628.

Selig, J. M., 2000. Clifford algebra of points, lines and planes. *Robotica*, 18(5), 545–556.

Senatore, J., Segonds, S., Rubio, W. and Dessein, G., 2012. Correlation between machining direction, cutter geometry and step-over distance in 3-axis milling: Application to milling by zones. *Computer-Aided Design*, 44(12), 1151–1160.

Shen, Y. H., Yao, X. H. and Fu, J. Z., 2011. Smooth non-uniform rational B-spline (NURBS) machining with kinematic limit for short linear segments. *International Journal of Computer Integrated Manufacturing*, 24(12), 1103–1116.

Shoemake, K., 1985. Animating rotation with quaternion curves. *ACM SIGGRAPH*, 19(3), 245–254.

Simpson, L. C. and Mullineux, G., 2009. Exponentials and motions in geometric algebra. In: *Proc. Computer Graphics, Computer Vision and Mathematics (GraVisMa)*, Skala, V. and Hildenbrand, D., eds., University of West Bohemia, Plzen, Czech Republic, 9–16.

Thomas, F., 2014. Approaching dual quaternions from matrix algebra. *IEEE Transactions on Robotics*, 30(5), 1037–1048.

Ting, K.-L. and Zhang, Y., 2004. Rigid body motion characteristics and unified instantaneous motion representation of points, lines, and planes. *Transactions of the ASME: Journal of Mechanical Design*, 126(4), 593–601.

Wang, X., Han, D., Yu, C. and Zheng, Z., 2012. The geometric structure of unit dual quaternion with application in kinematic control. *Journal of Mathematical Analysis and Applications*, 389(2), 1352–1364.

Wareham, R. and Lasenby, J., 2008. Mesh vertex pose and position interpolation using geometric algebra. In: *Articulated Motion and Deformable Objects*, Perales, F. J. and Fisher, R. B., eds., Lecture Notes in Computer Science, 5098, Springer-Verlag, Berlin, 122–131.

Wu, W. and You, Z., 2010. Modelling rigid origami with quaternions and dual quaternions. *Proceedings of the Royal Society, A*, 466(2119), 2155–2174.

Yang, J. and Altintas, Y., 2013. Generalized kinematics of five-axis serial machines with non-singular tool path generation. *International Journal of Machine Tools & Manufacture*, 75(2), 119–132.

Zhang, W., Zhang, Y. F. and Ge, Q. J., 2005. Interference-free tool path generation for 5-axis sculptured surface machining using rational Bézier motions of a flat-end cutter. *International Journal of Production Research*, 43(19), 4103–4124.

Zhang, Y. Xu, X. and Liu, Y. X., 2011. Numerical control machining simulation: a comprehensive survey. *International Journal of Computer Integrated Manufacturing*, 24(7), 593–609.

Zhu, L., Ding, H. and Xiong, Y., 2012. Simultaneous optimization of tool path and shape for five-axis flank milling. *Computer-Aided Design*, 44(12), 1229–1234.

Appendix A. Related approaches

This appendix presents related ideas and formulations of geometric algebra. The purpose is to show approaches used by others and to provide some motivation for the definition of the $\mathcal{G}_4$ algebra used in the paper. Note that the notation used in some approaches looks similar to that in others, but may have a significantly different meaning.

Many of the ideas go back to the 1800s and the work of Clifford, Grassmann and Hamilton to represent transformations of three (and higher) dimensional space. They were successful in this. However the introduction and popularization of matrix methods overtook their work.

It is straightforward to use vectors to represent points in three-dimensional space. Then 3×3 matrices can be applied by multiplication to generate rotations. The introduction of a fourth (homogeneous) coordinate allows three-dimensional space to be represented projectively. This in turn permits the use of 4×4 matrices to represent rotations and translation, and hence rigid-body transforms, in a single form (Röschel 1998).

Quaternions

The advent of computer graphics and computer games produced renewed interest in Hamilton's quaternions as a means of representing rotations of three-dimensional space. This was seen as a more robust approach. The ring of quaternions is formed by introducing three square roots of -1 to the field of real numbers. These are: i, j, k. The general quaternion is a linear combination of these and unity. Three-dimensional space is represented within the quaternions with the following map which embeds the typical point (Yang and Altintas 2013).

$$(x, y, z) \mapsto p = xi + yj + zk \tag{A1}$$

If (u, v, w) are the direction cosines of a line through the origin and θ is an angle then a rotation about the line is generated by the quaternion

$$R \;=\; (\cos \tfrac{1}{2}\theta) + (ui + vj + wk)(\sin \tfrac{1}{2}\theta) \tag{A2}$$

The transform is performed by mapping the typical point p to the product $Rp\overline{R}$ where $\overline{R}$ is the conjugate of R obtained by changing the signs of i, j, k (Yang and Altintas 2013).

One of the attractions of quaternions (for computer games and similar applications) is the ability to generate motions by interpolating between two rotations using spherical linear interpolation (slerp) (Shoemake 1985).

Double quaternions

In their pure form, quaternions cannot deal with translations. However two extensions permit this; these are double quaternions and dual quaternions.

A 4×4 matrix can represent a rigid-body transformation. It can be expressed as the product of two matrices, one representing a rotation about an axis through the origin, the other representing a translation (Röschel 1998). The translation matrix can be approximated by an orthogonal matrix which depends upon a large number R; this can be regarded as treating the translation as a rotation about a distant axis with R

being the radius (Etzel and McCarthy 1999). This means that the original matrix can be approximated by a 4×4 orthogonal matrix.

This allows the matrix to be expressed as the product of two matrices which correspond to quaternions g_1, g_2 (Etzel and McCarthy 1999). To help deal with such pairs, the idea of a double quaternion is introduced. This is a combination of the form $p + qe$ where e is a symbol which commutes with the quaternions and has the property that $e^2 = 1$ (Thomas 2014). If the dual numbers ξ and η are defined (Etzel and McCarthy 1999) by $\xi = \frac{1}{2}(1 + e)$ and $\eta = \frac{1}{2}(1 - e)$, then the quaternions g_1, g_2 are combined to form the double quaternion $G = g_1\xi + g_2\eta$ which represents the original transform. The two parts of G behave independently, and this allows combinations of transforms to be formed and manipulated. In particular, given control poses which are double quaternions, it is possible to form Bézier and B-spline combinations of the individual parts separately to form a changing transform and hence a motion (Etzel and McCarthy 1999, Ahlers and McCarthy 2001).

Dual quaternions

The other extension to quaternions are the dual quaternions. These are elements of the form $p + q\varepsilon$ where p and q are quaternions and ε is an additional symbol, added in the same way as e with double quaternions, whose square is zero: $\varepsilon^2 = 0$ (Thomas 2014)

The dual quaternions contain a copy of three-dimensional space given by the mapping

$$(x, y, z) \mapsto p = 1 + (xi + yj + zk)\varepsilon \tag{A3}$$

It is upon such elements that other dual quaternions act to represent transforms. If Q is a dual quaternion, it defines a map $p \mapsto Qp\overline{Q}$, where the bar denotes a conjugate given by $\overline{p + q\varepsilon} = \overline{p} - \overline{q}\varepsilon$. In particular, if (u, v, w) is a unit vector along an axis through the origin, then the quaternion R given by equation (A2), regarded as a dual quaternion, generates a rotation through an angle θ about the axis. The dual quaternion

$$T \;=\; 1 + \tfrac{1}{2}(t_1 i + t_2 j + t_3 k)\varepsilon \tag{A4}$$

generates a translation along the vector (t_1, t_2, t_3). For both R and T, the result of the map is again a point, and, since $R\overline{R} = T\overline{T} = 1$, the presence of unity as the part not involving ε in equation (A3) is preserved.

As with other formulations, combining control points, given as dual quaternions, using Bézier and B-spline techniques can be used to create motions along free-form curves or across free-form surfaces (Purwar and Ge 2005, Zhang *et al.* 2005, Purwar *et al.* 2008).

Clifford algebra

A Clifford (or geometric algebra) is formed from a vector space (Azmy 2013, Selig 2000). A multiplication is imposed on the elements of a basis which is then extended to the whole space. Key to the definition is the scalar (real) value assigned to the square of each basis vector and this gives rises to different forms of algebra. In fact it is only the sign of the square value that is important since it is straightforward to replace any basis vector by a scalar multiple of itself. The Clifford algebra $\mathrm{G}_{p,q,r}$ is one where p of the basis vectors square to $+1$, q of them square to -1, and r to zero, where $p + q + r$ is the full dimension.

Conformal geometric algebra

This is the most widely used form of geometric algebra being employed in applications such as computer vision (Dell'Acqua *et al.* 2008) and robotics (Bayro-Corrochano and Zamora-Esquivel 2006).

There are various forms and the one commonly used to handle three dimension geometry is $G_{3,1,0}$. Following notation combined from (Dorst 2010) and (Cibura and Dorst 2011), suppose the basis vectors are e_+, e_1, e_2, e_3, e_- with $e_+^2 = e_1^2 = e_2^2 = e_3^2 = 1$, and $e_-^2 = -1$. In addition, define $e_0 = \frac{1}{2}(e_- + e_+)$, and $e_\infty = e_- - e_+$.

Three dimensional space is included in the algebra using the map (Cibura and Dorst 2011)

$$(x, y, z) \mapsto p = \alpha[e_0 + (xe_1 + ye_2 + ze_3) + \tfrac{1}{2}(x^2 + y^2 + z^2)e_\infty] \tag{A5}$$

This is a null vector in the sense that under the multiplication $p^2 = 0$. The factor α is regarded as a homogeneous coordinate meaning that p is a projective representation of the original point.

If an axis through the origin has direction cosines (u, v, w), then the following element generates a rotation (Dorst 2010)

$$R \;=\; (\cos\tfrac{1}{2}\theta) - (\sin\tfrac{1}{2}\theta)B \;=\; \exp(-\tfrac{1}{2}B\phi) \tag{A6}$$

where $B = ue_2e_3 + ve_3e_1 + we_1e_2$. It is applied using the map $p \mapsto Rp\overline{R}$ where $\overline{R}$ denotes the reverse of R which can be considered, in this case, as its multiplicative inverse (Dell'Acqua *et al.* 2008). A translation along the vector (t_1, t_2, t_3) is created by the following element (Dorst 2010)

$$T \;=\; 1 - \tfrac{1}{2}te_\infty \;=\; \exp(-\tfrac{1}{2}te_\infty) \tag{A7}$$

applied in the same way, where $t = t_1e_1 + t_2e_2 + t_3e_3$.

As is seen, the above transforming elements can be written as exponentials of other elements within the algebra. This means that given control poses in exponential form, a motion can be generated by forming Bézier and B-spline combinations of the exponents (Wareham and Lasenby 2008).

$\mathcal{G}_4$

The geometric algebra $G_{0,3,1}$ has been used successfully in applications such as robotics, rigid-body motion and computer vision (Azmy 2013, Ting and Zhang 2004, Selig 2000). It creates a model of Euclidean geometry in which planes in geometry are represented by vectors in the algebra and points in geometry correspond to elements in the algebra of grade 3 (trivectors). This seems the wrong way round: it seems more natural for points to corresponds to vectors, and planes to trivectors. This is the motivation for the definition of $\mathcal{G}_4$. In $G_{0,3,1}$, one of the basis vectors squares to zero, call this vector e_0. One might hope to obtain more natural correspondences by "reversing" the definition and have e_0 square to infinity. Clearly this is difficult to handle computationally and it leads to the use of the symbol ε which represents a small number. This enables e_0^2 to be defined as ε^{-1} as in equation (4) (Mullineux 2002, 2004). For convenience as much as anything else, the squares of the other three basis vectors are taken to be $+1$ rather than -1

(equation (3). The quantity ε needs to be carried (symbolically) through any calculation and then, if necessary (for example when plotting a pose), is allowed to become zero. Effectively, the coefficients in the typical element, as in equation (1), are power series in ε (Mullineux and Simpson 2011). In a computer implementation these are represented by arrays (of fixed length) of coefficients. Algebraic operations are performed on these arrays in a term-by-term manner with excess coefficients being allowed to "drop off' the end of the array.

The algebra provides a projective model of three-dimensional space with points mapping as

$$(x, y, z) \mapsto p = W(e_0 + xe_1 + ye_2 + ze_3) \tag{A8}$$

where W is a homogeneous coordinate. It is possible (Mullineux 2002, 2004) to define the products in equations (6) and (7) for all elements in the algebra; in particular, the outer product is always a binary product of pairs of elements. This is unconventional compared to other geometric algebras and the consequent tests for collinearity and coplanarity are similar but different.

If S is an even-grade element, then the map $x \mapsto \overline{S}xS$ sends vectors to vectors (cf. discussion before equation (8)) and generates a rigid-body transform of three-dimensional space (Mullineux 2004). Any pseudo-scalar generates the identity transform, and the subspace of non-zero, even-grade elements generates all possible rigid-body transforms. If (u, v, w) is a unit vector through the origin then the even-grade element

$$R \;=\; (\cos \tfrac{1}{2}\theta) + (\sin \tfrac{1}{2}\theta)B \;=\; \exp(\tfrac{1}{2}B\theta) \tag{A9}$$

produces a rotation about the axis through an angle θ. Similarly, the element

$$T \;=\; 1 + \tfrac{1}{2}e_0 t \;=\; \exp(\tfrac{1}{2}e_0 t) \tag{A10}$$

creates a translation along the vector (t_1, t_2, t_3) where $t = t_1e_1 + t_2e_2 + t_3e_3$. The fact that a non-zero even-grade element provides a rigid-body transform means that a non-zero linear combination of even-grade elements again gives a transform. This means that Bézier and B-spline combinations can be formed additively as well as multiplicatively (as in section 4.3) provided no intermediate control pose becomes zero.

List of figures

1	typical_motion.eps	Example of an L-shaped block with the path followed by one vertex
2	lin_int.eps	Linear MI (translation) between two given tool poses
3	cir_int.eps	Circular MI between two given tool poses
4	ff_int_slerp.eps	Free-form MI between two given tool poses with two additional control poses (shown dashed) as given in table 2
5	linear_twist.eps	Linear MI with twist between two given tool poses
6	cir_int_twist_slerp.eps	Circular MI with twist between two given tool poses
7	typical_screw.eps	Typical screw motion as a slerp interpolation between $S_0 - 1$ and $S_1 = U$ using equation (9): $S(t) = U^t$ for $0 \leq t \leq t$
8	rot_all.eps	Three examples of linear MI – top: no rotation about vertical axis, middle: equal rotations, bottom: six cases of different rotations
9	screw_example_both_all.eps	Example of positive and negative screw motions
10	picture_bicubic_with_axes_flat.eps	Grid of precision points and poses (for $n = 5$) on a Bézier surface
11	bezier_biquadratic_flat.eps	Rational Bézier biquadratic patch
12	bezier_biquadratic_error_plots.eps	Error plots for rational Bézier biquadratic patch
13	bezier_bicubic_flat.eps	Bézier bicubic patch
14	bezier_bicubic_error_plots.eps	Error plots for Bézier bicubic patch